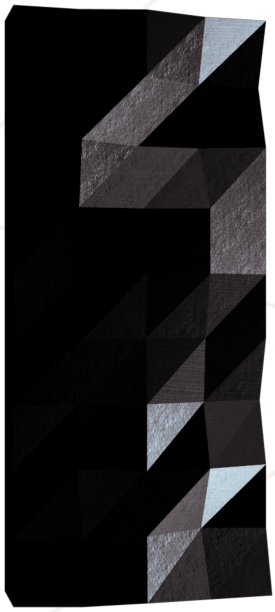

# C Piscine

## Rush 02

*Summary: This document is the subject for Rush02 of the C Piscine @ 42.*

*Version: 9*

# Contents

# Chapter I

# Instructions

- The group WILL be registered to defense. Automatically. If one of of you cancels it, there won't be another one.
- Any question regarding the subject will probably complicate the subject.
- You have to follow the submission procedures for all your exercises.
- This subject could change up to an hour before submission.
- Moulinette compiles with the following flags: -Wall -Wextra -Werror; and uses `gcc`.
- If your program doesn't compile, you'll get `0`.
- You must turn in a Makefile, which will compile your project using the rules `$NAME`, `clean` and `fclean`
- You must therefore do the project with the imposed team and show up at the defense slot you've assigned, with all of your teammates.
- Each member of your group must be fully aware of the works of the project. Should you choose to split the workload, make sure you all understand what everybody's done. During defense, you'll be asked questions, and the final grade will be based on the worst explainations.
- Gathering the group is your responsibility. You've got all the means to get in contact with your teammates: phone, email, carrier pigeon, spiritism, etc. So don't bother blurping up excuses. Life isn't always fair, that's just the way it is.
- However, if you've really tried everything one of your teammates remains unreachable : do the project anyway, and we'll try and see what we can do about it during defense. Even if the group leader is missing, you still have access to the submission directory.
- It goes without saying, but your work must respect the Norm. Be thourough. And enjoy !

# Chapter II

# Foreword

Here is a old-fashioned pecan pie recipe for you :

```
Ingredients
- Pastry dough
- 3/4 stick unsalted butter
- 1 1/4 cups packed light brown sugar
- 3/4 cup light corn syrup
- 2 teaspoon pure vanilla extract
- 1/2 teaspoon grated orange zest
- 1/4 teaspoon salt
- 3 large eggs
- 2 cups pecan halves (1/2 pound)
Accompaniment: whipped cream or vanilla ice cream

Preparation:
Preheat oven to 350°F with a baking sheet on middle rack.
Roll out dough on a lightly floured surface with a lightly floured rolling pin
into a 12 inch round and fit into a 9 inch pie plate.
Trim edge, leaving a 1/2-inch overhang.
Fold overhang under and lightly press against rim of pie plate, then crimp
decoratively.
Lightly prick bottom all over with a fork.
Chill until firm, at least 30 minutes (or freeze 10 minutes).
Meanwhile, melt butter in a small heavy saucepan over medium heat.
Add brown sugar, whisking until smooth.
Remove from heat and whisk in corn syrup, vanilla, zest, and salt.
Lightly beat eggs in a medium bowl, then whisk in corn syrup mixture.
Put pecans in pie shell and pour corn syrup mixture evenly over them.
Bake on hot baking sheet until filling is set, 50 minutes to 1 hour.
Cool completely.

Cooks notes:
Pie can be baked 1 day ahead and chilled. Bring to room temperature before
serving.
```

# Chapter III

# Subject

|  | Exercise 00 |
|---|---|
| rush-02 | |
| Turn-in directory : $ex00/$ | |
| Files to turn in : `Makefile and all the necessary files` | |
| Allowed functions : `write, malloc, free, open, read, close` | |

- Create a program that takes a number as argument and converts it to its written letters value.
- Executable name: `rush-02`
- Your source code will be compiled as follows :

```
make fclean
make
```

- Your program can take up to 2 arguments:
  - If there is only one argument, it is the value you need to convert
  - If there are two arguments, the first argument is the new reference dictionary and the second argument is the value you need to convert.
- If the argument isn't a valid unsigned integer, your program must return "Error", followed by a newline.
- Your program must parse the dictionary given as resource to the project. The values inside it must be used to print the result. These values can be modified.
- Any memory allocated on the heap (with malloc(3)) must be freed correctly. This will be verified during evaluation.
- The dictionary will have the following rules:

```
[a number][0 to n spaces]:[0 to n spaces][any printable characters]\n
```

- Numbers are to be handled the same way atoi handles them.
- You will trim the spaces before and after the value in the dictionary.
- The dictionary will always have at least the keys as in the reference dictionary. Their value can be modified, more entries can be added, but the initial keys can't be removed.
- You only need to use the initial entries (For instance, if we add `54: fifty-four`, you still have to use `50: fifty` and `4: four`)
- The entries of the dictionary can be stored in any order.
- There can be empty lines in the dictionary.
- If you have any errors from the dictionary parsing, your program must output "Dict Error\n"
- If the dictionary doesn't allow you to resolve the asked value, your program must output "Dict Error\n".

- Example:

```
$> ./rush-02 42 | cat -e
forty two$
$> ./rush-02 0 | cat -e
zero$
$> ./rush-02 10.4 | cat -e
error$
$> ./rush-02 100000 | cat -e
one hundred thousand$
$> grep "20" numbers.dict | cat -e
20    :     hey     everybody !$
$> ./rush-02 20 | cat -e
hey everybody !$
```

# Chapter IV

# Bonus

- Using `-`, `,`, `and` to be closer to the correct written syntax
- Doing the same exercice in a different language. For this, you are allowed to provide another dictionary which will contain the necessary entries.
- Using `read` to read standard entry when there is no argument